AF IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Robert T. George, et al.	§	Group Art Unit:	2181
		§		
Serial No.:	10/630,286	§		
		§	Examiner:	Niketa I. Patel
Filed:	July 30, 2003	§		
		§		
For:	Associating Address Space Identifiers With Active Contexts	§	Atty. Dkt. No.:	ITL.1034US (P16844)
		§		

Mail Stop **Appeal Brief-Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDED APPEAL BRIEF TRANSMITTAL

Sir:

Transmitted herewith is the Amended Appeal Brief for this application. The First Appeal Brief was filed on October 25, 2006

Pursuant to M.P.E.P. § 1205, there is no fee due for this Appeal, because the Examiner requested that the Appellant file a new Appeal Brief in compliance with 37 CFR 41.37(c) after filing of the first Appeal Brief on October 25, 2006. The Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 20-1504.

Respectfully submitted,

Date: 2/2/07

Mark J. Rozman
Registration No. 42,117
TROP, PRUNER & HU, P.C.
1616 S. Voss Road, Suite 750
Houston, Texas 77057-2631
(512) 418-9944 [Phone]
(713) 468-8883 [Fax]
Customer No.: 21906

Date of Deposit: 2-2-07
I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Stephanie Petreas

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Robert T. George, et al.	§	Group Art Unit:	2181
		§		
Serial No.:	10/630,286	§		
		§	Examiner:	Niketa I. Patel
Filed:	July 30, 2003	§		
		§		
For:	Associating Address Space Identifiers With Active Contexts	§ §	Atty. Dkt. No.:	ITL.1034US (P16844)

Mail Stop **Appeal Brief-Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDED APPEAL BRIEF

Date of Deposit: 2-2-07

I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

Stephanie Petreas

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST 3

II. RELATED APPEALS AND INTERFERENCES 4

III. STATUS OF CLAIMS 5

IV. STATUS OF AMENDMENTS 6

V. SUMMARY OF CLAIMED SUBJECT MATTER 7

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL 14

VII. AGRUMENT 15

VIII. CLAIMS APPENDIX 20

IX. EVIDENCE APPENDIX 23

X. RELATED PROCEEDINGS APPENDIX 24

I. REAL PARTY IN INTEREST

The real party in interest is the assignee Intel Corporation, the assignee of the present application by virtue of the assignments recorded at Reel/Frame 014354/0922 and 015014/0304.

II. RELATED APPEALS AND INTERFERENCES

This appeal may relate to the co-pending appeal in U.S. Patent Application No. 10/630,287 filed July 30, 2003 entitled "Dynamically Partitioning Pipeline Resources" in the names of Robert T. George, Jason W. Brandt, K. S. Venkatraman, and Sangwook P. Kim.

III. STATUS OF CLAIMS

Claims 1, 3, 5-14, 16-17, 20, 22-25, 27, and 29-33 stand rejected. Claims 2, 4, 15, 18-19, 21, 26, and 28 have been cancelled. The rejections of all pending claims 1, 3, 5-14, 16-17, 20, 22-25, 27, and 29-33 are being appealed.

IV. STATUS OF AMENDMENTS

Appellants filed a "Reply to Final Office Action Mailed May 31, 2006" on July 25, 2006 that included an amendment to claim 31. In an Advisory Action dated August 11, 2006, the Examiner indicated that the amendment would not be entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

At this point, no issue has been raised that would suggest that the words in the claims have any meaning other than their ordinary meanings. Nothing in this section should be taken as an indication that any claim term has a meaning other than its ordinary meaning.

Independent claim 1 is an independent apparatus claim. Support for the subject matter of independent claim 1 can be found, for example, in the Specification as filed at p. 3, ln. 13 – p. 5, ln. 22, and p. 18, lns. 11-19.

Independent claim 8 is an independent method claim. Support for independent claim 8 may be found, for example, in the Specification at p. 4, ln. 19 – p. 5, ln. 6, and p. 7, ln. 15 – p. 9, ln. 4.

Claim 16 is an independent system claim. Support for claim 16 may be found, for example, at p. 3, ln. 13 – p. 5, ln. 22, and p. 4, ln. 19 – p. 5, ln. 6, and p. 7, ln. 15 – p. 9, ln. 4 and p. 18, lns. 11-19. Further support for claim 16 may be found, for example, at p. 26, ln. 3 – p. 27, ln. 21.

Claim 20 is an independent article claim. Support for claim 20 may be found, for example, at p. 3, ln. 13 – p. 5, ln. 22, and p. 18, lns. 11-19. Further support for claim 20 may be found at p. 25, ln. 14 – p. 26, ln. 2.

Claim 25 is an independent method claim. Support for claim 25 may be found, for example, at p. 4, ln. 19 – p. 5, ln. 6, and p. 7, ln. 15 – p. 9, ln. 4.

Support for dependent claim 5 may be found, for example, in the Specification as filed at p. 5, lns. 8-10.[1]

Support for dependent claim 11 may be found, for example, in the Specification as filed at p. 5, lns. 11–26.

Support for dependent claim 22 may be found, for example, in the Specification as filed at p. 5, lns. 8–10.

[1] It is respectfully noted that the Examiner required the filing of this Amended Appeal Brief, as the Examiner stated that "summary of claimed subject matter fails to map the dependent claims…that are argued separately, to the specification by page and line number." Notice of Non-Compliant Appeal Brief, mailed January 8, 2007. However, 37 C.F.R. §41.37(c)(1)(v) nowhere requires such concise explanation of dependent claims when they are not in means-plus-function format. 37 C.F.R. §41.37(c)(1)(v). Nevertheless, in the interest of moving this Appeal Brief forward, Applicants provide the requested information herewith.

Support for dependent claim 31 may be found, for example, in the Specification as filed at p. 6, lns. 1–18.

The following is a further concise explanation of the subject matter defined in the independent claims of the present appeal, also identified by page and line number in the specification.

In various embodiments of the present invention, pipeline resources or structures may be dynamically partitioned to provide support for multiple address spaces. By "dynamically," it is meant that the partitioning of pipeline resources may be modified during operation such that at different times, different address spaces may occupy a given partition of a pipeline resource and/or that different partitions may consume more or less of the resource. As used herein, the term "address space" means a set of addresses in memory corresponding to a given application (e.g., a context). Specification, p. 3, lns. 13–24.

Various embodiments may also include a store filter (also termed a "flush filter" herein) which monitors stores for updates to page tables, and may selectively invalidate associated translations. address space identifiers (ASIDs) or address space numbers (ASNs) may be used to augment linear addresses in various pipeline resources with a pointer to the context with which they are associated. An "address space identifier" or "address space number" may be any number, code, or other notation which identifies one or more address spaces with which it is associated. This allows multiple application contexts to share pipeline structures, reducing context-switch overhead. For example, when a context switch occurs, the current ASID value is changed, rather than flushing the pipeline structures. Similarly, in certain embodiments, a thread identifier (thread ID) may be provided to identify a given processor thread for a corresponding address space. Specification, p. 4, ln. 10 – p. 5, ln. 6.

In certain embodiments, the flush filter may monitor updates to intermediate page table entries, and selectively flush TLB entries corresponding to a specific ASID. While discussed herein primarily with respect to TLB's, it is to be understood that embodiments may be applied to any pipeline resource. Various architectural events may cause a selective flush of the TLBs in accordance with embodiments of the present invention. Specification, p. 5, lns. 7-14.

In various embodiments, ASIDs may be used to augment the linear address in pipeline resources with a pointer to the corresponding address space. In such embodiments, the microprocessor may maintain a global current ASID register (or ASID manager) that is updated when a new address space is created or when changing to a different, previously seen address space. TLB insertions may be extended with the current ASID value, and TLB lookups match only if the ASID tag matches the current ASID value. When a context switch triggers an address space change, the microprocessor may switch to a different ASID value that represents the new address space, instead of flushing the TLB's and other pipeline structures. In certain embodiments, selectively flushing entries corresponding to a specific address space may provide a substantial performance gain for environments with multiple contexts. Specification, p. 5, ln. 24 – p. 6, ln. 14.

In one embodiment, ASIDs may be implemented using a two-bit ASID (i.e., four address space contexts) per thread. Various pipeline resources may be augmented with ASIDs, which may be used to selectively invalidate entries based on the ASID value. A trace cache mini-tag may also be augmented with an ASID, as several TC traces may alias in a typical linear address region, differing by ASID value. For an iTLB several translations may alias in the same linear address region but differ by ASID. In such an embodiment, the iTLB may allow invalidation of specified ASIDs via various flush mechanisms. These mechanisms may include invalidating entries by a specific address with a given ASID value; invalidating all non-global entries; invalidating all entries for a given ASID (global and non-global); and invalidating all entries. In one embodiment, ASIDs may be assigned on a per thread basis, thus expanding the scope of ASIDs. Specification, p. 6, ln. 15 – p. 7, ln. 14.

In one embodiment, a hashing technique may be used to hash the two most significant bits (MSBs) of a pipeline resource with the two-bit ASID value. In such manner ASID values may be implemented with virtually no area growth. In one embodiment a branch target buffer (BTB) may obtain the correct ASID from a next instruction (NIP) control register using a thread ID. The target array may look at two bits to determine its behavior, a target array Exclusive Or (XOR) disable bit; and a target array ASID disable bit. In such an embodiment, when ASIDs are enabled, the XOR feature may be enabled and the ASID bits may be hashed (XOR'ed) into the upper two MSBs of the target tag. When ASIDs are enabled but the XOR feature disabled, the target tag may be ten bits of "tag" (least significant bits (LSBs)) and two-bits of ASID (MSB).

In one embodiment the global array tag may include 512 sets, 4-way set associative. The field in each way may include a valid (two bits); an offset (4 bits); a tag (4 bits), a counter (two bits) and a least recently used (LRU)/set (3 bits). In such an embodiment, the global array may look at a global array XOR/ASID disable bit. As above, when ASIDs and XOR are enabled the ASID bits may be hashed into the two MSBs of the tag. Specification, p. 6, ln. 15 – p. 8, ln. 19.

In one embodiment, a flush filter may be implemented as a content addressable memory (CAM) which matches post-retirement store addresses (i.e., physical addresses) against known page directory base and page table bases to determine if the store targets a page table belonging to one of the ASID contexts. If the flush filter finds a match, the store is attempting to update a page table entry cached in the TLB, and the flush filter provides the corresponding ASID value and thread. The associated ASID partition may then be marked as hit/modified, and the partition flush may be delayed until the next TLB flushing event. As a result, the flush filter identifies all page table updates that would change a currently cached TLB entry, and entries corresponding to a specific address space are invalidated. The resulting ASID flush (essentially a selective TLB flush) may be delayed until a context switch (i.e., a next TLB flushing event). Specification, p. 8, ln. 25 – p. 9, ln. 14.

In various embodiments, during steady state operation (i.e., after the flush filter has already been filled), the flush filter may monitor updates to page tables by filtering addresses of senior stores and external snoops. The physical address bits of these operations may be CAM'ed against the base address stored in the flush filter. The flush filter may be filled during TLB page walks. On every page walk, addresses representing page directory bases and page table bases, along with the current thread ID and ASID, may be sent to the flush filter. For each address sent to the flush filter during the fill, it may check if that address is already stored for the same (current) Thread and ASID context. If there is not an exact match (including the case where the physical address matches but the thread ID or ASID do not), then the flush filter allocates a new entry in the CAM with the address as the tag, and the thread ID and ASID as the match data, regardless of whether a global bit is set, meaning the page translation is used in another context. Specification, p. 10, ln. 10 – p. 11, ln. 9.

Referring now to FIG. 1B, shown is a block diagram of a flush filter in accordance with one embodiment of the present invention. As shown in FIG. 1B, a portion of a processor pipeline includes a DAC multiplexer 15, a filter 20, a store buffer 30, a flush vector control register (or "flush vector") 35, a page miss handler (PMH) 40, and a second multiplexer 45. Flush vector 35 may receive hits of a CAM match (i.e., tag address, ASID, and thread ID) via line 22 from filter 20. As shown in the embodiment of FIG. 1B, each entry of filter 20 may include a first valid bit (V0), a tag address. Additionally, the entry may include a second valid bit (V1). In certain embodiments, the presence of two valid bit arrays may be used to handle filter fill operations while a particular partition of the filter is being invalidated. Entries may further include an ASID and a thread ID, which are shown as N-bit and M-bit numbers, respectively. As shown in FIG. 1B, tag addresses may be stored in filter 20 via line 25 which is received in a fill port of filter 20. Specification, p. 11, ln. 10 – p. 12, ln. 24.

As further shown in FIG. 1B, a multiplexer 45 may be coupled to filter 20 via line 28 to provide signals to an invalidate port of filter 20. In one embodiment, microcode instructions may be used to invalidate entries of filter 20. Specification, p. 13, lns.6-10.

Referring now to FIG. 2, shown is a block diagram of flush filter system in accordance with one embodiment of the present invention. As shown in FIG. 2, a flush filter 20 may receive tag addresses from page miss handler 40. For each entry filled into filter 20, an ASID manager 45 may provide a current ASID for the appropriate address space. ASID manager 45 may store various CR3 values and provide an associated ASID for a present context.

When flush filter 20 compares senior store addresses or external snoops to tag addresses stored therein and a CAM match is found, a hit for the matched entry in flush vector 35 may be provided to flush filter controller 60. Flush filter controller 60 may be used to provide invalidate instructions to flush filter 20, and to send flush instructions to associated TLBs. As shown in FIG. 2, these TLBs may include an iTLB 50, a dTLB 70 and a STLB 75. Of course, these TLBs may have extended entries that include ASIDs and thread IDs. Specification, p. 13, ln. 22 – p.14, ln. 5.

Referring now to FIG. 3, shown is a flow diagram of a method in accordance with one embodiment of the present invention. As shown in FIG. 3, the method begins by obtaining a physical address (block 110). Next, the filter may compare the physical address obtained to tag addresses in the filter (block 120). Such comparison may be performed to determine whether there is a match between the physical address and any of the tag addresses stored in the filter and corresponding ASID and thread ID (diamond 130). If no such matches occur, control returns to block 110 where another physical address may be obtained. If one or more matches is found, valid bit V0 may be set if there is no conflicting invalidating access occurring, and valid bit V1 may be set if there is a conflicting invalidating access occurring. Specification, p. 14, lns. 6–25.

If there is a match, a control register coupled to the filter may be updated (block 140). Still referring to FIG. 3, the address space or spaces of the filter corresponding to the updated address spaces in flush vector 35 may be invalidated (block 150). In certain embodiments, microcode may read the contents of flush vector 35 and determine that a particular ASID and thread ID have been set and perform an invalidation process. In such a process, microcode may cause the V0 array to be invalidated at a cycle n and the V1 array to be invalidated at a cycle n+1. Further, microcode may cause flush vector 35 to be cleared. Thus for a given ASID and thread ID, when an update is detected by filter 20, entries in filter 20 corresponding to the particular ASID and thread ID may be invalidated. Finally, on the next context switch the invalidated address space may be flushed from the filter and associated pipeline structures, such as the TLB (block 160). Specification, p. 15, ln. 12 – p. 16, ln. 8.

Referring now to FIG. 4, shown is a block diagram showing correspondence between a flush filter and a TLB in accordance with one embodiment. As shown in FIG. 4, flush filter 20 includes physical addresses (tag addresses) and associated valid, thread ID and ASID bits. Filter 20 may be coupled to an associated TLB 50 which is similarly augmented to include the valid bit, thread ID, and ASID, along with corresponding linear and physical addresses. Specification, p. 18, lns. 11-19.

Since there are a large number of possible base combinations for a single address space, in certain embodiments the flush filter may be partitioned to prevent a greedy application from consuming all the flush filter match slots. Two flush filter partitioning mechanisms may be applied to prevent ASID capacity issues: static partitioning and dynamic partitioning, both of which may provide for overflow conditions. In certain embodiments, the flush filter may be segmented with a static partition per ASID. In an embodiment having a 32-entry CAM, a hard-partitioned flush filter with a 2-bit ASID per thread ID would allow 4 CR3/PD base/PT base entries per ASID/thread ID. In an embodiment having a 32-entry CAM with dynamic partitioning, a 2-bit ASID per thread ID would allow any ASID/thread ID context to consume a variable number of CR3/PD base/PT base entries. Specification, p. 21, lns. 3-19.

Referring now to FIG. 5, shown is a block diagram of a representative multiprocessor computer system 200 in accordance with one embodiment of the invention. As shown in FIG. 5, multiprocessor computer system 200 includes a first processor 201 and a second processor 211.

Processors 201 and 211 may be coupled over a front-side bus 220 to a memory hub 230 in one embodiment, which may be coupled to a shared main memory 240 via a memory bus.

Memory hub 230 may also be coupled (via a hub link) to an input/output (I/O) hub 235 that is coupled to an I/O expansion bus 255 and a peripheral bus 250. In various embodiments, I/O expansion bus 255 may be coupled to various I/O devices such as a keyboard and mouse, among other devices. Peripheral bus 250 may be coupled to various components such as peripheral device 270 which may be a memory device such as a flash memory, add-in card, and the like. Specification, p. 26, lns. 3-19.

As shown in FIG. 5, first processor 201 may include a TLB 203 and a filter 205 in accordance with an embodiment of the present invention. More so, a level 2 (L2) cache 207 may be coupled to processor 201. Similarly, second processor 211 may include a TLB 213, a filter 215 and may be coupled to a L2 cache 217. Specification, p. 26, ln. 22 – p. 27, ln. 1.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Each of the following grounds of rejection are presented for review:

(1) claims 1, 3, 6 and 7 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(2) claim 5 stands rejected under 35 U.S.C. §102(e) over Zuraski;

(3) claims 8-10, 12-14, 25, 27, 29-30, and 33 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(4) claim 11 stands rejected under 35 U.S.C. §102(e) over Zuraski;

(5) claims 16 and 17 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(6) claim 31 stands rejected under 35 U.S.C. §102(e) over Zuraski;

(7) claims 20, 23, 24 and 32 stand rejected under 35 U.S.C. §102(e) over Zuraski;

(8) claim 22 stands rejected under 35 U.S.C. §102(e) over Zuraski;

VII. ARGUMENT

(1) Claims 1, 3, 6 and 7 Are Patentable Under 35 U.S.C. §102(e) over Zuraski

Independent claim 1 is an apparatus that recites a pipeline resource that includes multiple address spaces each corresponding to one of multiple address space identifiers, where the pipeline resource includes entries each including one of the address space identifiers and where the entries are selectively flushable on an address space basis. Claim 1 stands rejected under 35 U.S.C. §102(e) over U.S. Patent No. 6,510,508 (Zuraski). This rejection is clearly erroneous, as Zuraski fails to teach all of the recited subject matter of claim 1 and thus anticipation cannot be established.

As to claim 1, Zuraski does not teach a pipeline resource including entries that are selectively flushable on an address space basis. Instead, the TLB 39 of Zuraski, contended to be the pipeline resource, does not include entries that are selectively flushable. Rather, as taught by Zuraski, the TLB is flushed in its entirety. The Examiner cites Zuraski, col. 9, lns. 52-55, however, all this teaches is a total flush of the TLB, not a selective flush on an address space basis ("In the embodiment shown, TLB flush filter may assert an Invalidate signal in order to allow a flush of TLB 39." Zuraski, col. 9, lns. 52-55.). Furthermore, the Examiner refers to col. 13, lns. 3-11. However, this portion similarly teaches that the entire TLB is flushed. Any doubt as to the teaching of Zuraski is easily resolved in Appellants' favor: Zuraski teaches a total flush of the TLB, nothing more nothing less: "a flush of TLB may occur when filter circuit 403 asserts an invalidate signal, thereby *invalidating all entries* currently stored in TLB 39." Zuraski, col. 11, lns. 14-17 (emphasis added).

Simply put, there is no teaching in Zuraski that either its flush filter or its TLB includes entries that are selectively flushable, and certainly not on an address space basis. Thus the rejection of claim 1 and its dependent claims is clearly erroneous and should be reversed.

(2) Claim 5 Is Patentable Under 35 U.S.C. §102(e) over Zuraski

Claim 5 depends from claim 1 and further recites that the entries in the pipeline resource further include a thread identifier. Claim 5 also stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper at least for the same reason discussed above regarding claim 1 (*see* VII.1.).

As support for this rejection of claim 5, the Examiner refers to column 9 of Zuraski, lines 43-47. However, all that this portion of Zuraski teaches is that a base address register (separate from the TLB of Zuraski) may store a base address of a page directory pointer table or other address information of a currently running context. Nowhere, however, does this or any other portion of Zuraski teach that entries in the TLB include a thread identifier in addition to an address space identifier. Accordingly, for this further reason the rejection of claim 5 is improper and should be reversed.

(3) Claims 8-10, 12-14, 25, 27, 29-30, and 33 Are Patentable Under 35 U.S.C. §102(e) over Zuraski

Independent claim 8 recites a method including associating an address space identifier with a value, hashing the address space identifier with a portion of the value, and thereafter storing the value and the address space identifier in a pipeline resource. Claim 8 stands rejected under 35 U.S.C. § 102(e) over Zuraski. As to claim 8, Zuraski nowhere teaches, at least, hashing an address space identifier with a portion of a value before storage of the value and the address space identifier.

The Examiner contends that the hashing recited in claim 8 is met by the teaching in col. 1, lns. 11-41 of Zuraski. However, this portion of the Background of Zuraski simply teaches that memory management systems translate virtual addresses into physical addresses. Nowhere, however, does Zuraski teach that such translations require execution of a hash on any numbers. Certainly Zuraski nowhere teaches that such translations are performed by hashing an address space identifier with a portion of a value associated with the address space identifier. Rather, all that Zuraski teaches in this cited portion is that virtual address-to-physical address translations, which may be obtained by "various mechanisms," are stored in a TLB. Simply put, nothing in Zuraski teaches hashing of address space identifiers with another value.

Thus the rejection of claim 8 and the claims depending therefrom is clearly erroneous. This is especially so, as this portion of Zuraski does not even disclose address spaces whatsoever, and certainly not address space identifiers. Thus claim 8 and the claims depending therefrom are patentable and for the same reason claim 25 and its dependent claims are patentable.

(4) Claim 11 Is Patentable Under 35 U.S.C. §102(e) over Zuraski

Claim 11 depends from, *inter alia*, claim 8 and further recites selectively flushing an entry of the pipeline resource after invalidating the entry. Claim 11 stands rejected under 35 U.S.C. §102(e) over Zuraski. The rejection of claim 11 is improper at least for the same reasons discussed above regarding claim 8 from which it depends (*see* VII.3.).

Furthermore, the rejection of claim 11 is improper as Zuraski nowhere teaches *selectively flushing* an entry after invalidation. Instead, Zuraski clearly teaches that the entire TLB is flushed after the corresponding flush filter 40 generates an Invalidate signal. Zuraski, col. 11, lns. 14-17. As taught in Zuraski, there is nothing selective about its flushing activities; the entire TLB is flushed. For this further reason, the rejection of claim 11 is improper and should be reversed.

(5) Claims 16 and 17 Are Patentable Under 35 U.S.C. § 102(e) over Zuraski

Claim 16 recites a system including, *inter alia,* a processor with a pipeline resource that includes multiple entries each having one of multiple address spaces, where each of the address spaces corresponds to one of multiple address space identifiers and a hashing engine to hash one of the address space identifiers with a portion of a value to be stored in one of the entries.

Claim 16 stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper as Zuraski nowhere teaches a hashing engine to hash an address space identifier with a portion of a value to be stored in an entry. Instead, all that the cited portions of Zuraski teach is a TLB that stores virtual-to-physical address translations (the Examiner simply refers to the background of Zuraski which teaches the existence of prior art TLBs that store virtual-to-physical address translations.).

Look carefully at Zuraski. There is no reference to hashing, and certainly there is no structure that is indicated to be a hashing engine, as recited by claim 16. Accordingly, the rejection of claim 16 and its dependent claims is clearly erroneous and should be reversed.

(6) Claim 31 Is Patentable Under 35 U.S.C. § 102(e) over Zuraski

Claim 31 depends from claim 16 and further recites a filter coupled to the pipeline resource to flush entries of one address space while entries of missing address spaces are maintained in the pipeline resource. Claim 31 stands rejected under 35 U.S.C. §102(e) over

Zuraski. This rejection is improper at least for the same reason discussed above regarding claim 16 from which it depends (*see* VII.5). Furthermore, the flush filter in Zuraski causes a flush of the entire TLB; it does not flush one address space while allowing other address spaces to be maintained in the TLB. Zuraski, col. 11, lns. 14-17. For this further reason the rejection of claim 31 is improper and should be reversed.

(7) Claims 20, 23, 24 and 32 Are Patentable Under 35 U.S.C. § 102(e) over Zuraski

Claim 20 is an independent article claim that includes instructions to associate an address space identifier with a value, store the value and the address space identifier in an entry of a pipeline resource, and flush a portion of the pipeline resource, where the portion includes the entry and other entries with the same address space identifier. Claim 20 stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper.

To validly anticipate claim 20, Zuraski would need to teach a pipeline resource that is capable of flushing only a portion of the resource, namely a portion that includes an entry storing a given address space identifier and other entries having that same address space identifier. Nothing in Zuraski, however, teaches a pipeline resource that can have only a portion flushed. Instead, the entire TLB of Zuraski is flushed, without regard to address space identifier. Zuraski, col. 9, lns. 50-58; col. 11, lns. 9-16; col. 13, lns. 3-11. Thus the rejection of claim 20 and its dependent claims is clearly erroneous.

(8) Claim 22 Is Patentable Under 35 U.S.C. § 102(e) over Zuraski

Claim 22 depends from claim 20 and further recites instructions to enable a system to store a thread identifier in an entry of a pipeline resource in addition to storing an address space identifier in the pipeline resource entry. Claim 22 stands rejected under 35 U.S.C. §102(e) over Zuraski. This rejection is improper for at least the same reason discussed above regarding claim 20 (*see* VII.7.) from which claim 22 depends.

As support for this rejection of claim 22, the Examiner refers to column 9 of Zuraski, lines 43-47. However, all that Zuraski teaches here is that a base address register (separate from the TLB of Zuraski) may store a base address of a page directory pointer table or other address information of a currently running context. Nowhere, however, does this or any other portion of Zuraski teach that entries in the TLB include a thread identifier stored in addition to an address space identifier. Accordingly, for this further reason the rejection of claim 22 is improper and should be reversed.

Appellants respectfully request that each of the final rejections be reversed and that the claims subject to this Appeal be allowed to issue.

Respectfully submitted,

Date: 2/2/07

Mark J. Rozman
Registration No. 42,117
TROP, PRUNER & HU, P.C.
1616 S. Voss Road, Suite 750
Houston, Texas 77057-2631
(512) 418-9944 [Phone]
(713) 468-8883 [Fax]
Customer No.: 21906

VII. CLAIMS APPENDIX

The claims on appeal are:

1. An apparatus comprising:

a pipeline resource having a plurality of address spaces, each of the plurality of address spaces corresponding to one of a plurality of address space identifiers, the pipeline resource including entries each including one of the plurality of address space identifiers, wherein the entries are selectively flushable on an address space basis.

3. The apparatus of claim 1, further comprising a control register coupled to the pipeline resource to provide the plurality of address space identifiers to the entries.

5. The apparatus of claim 1, wherein the entries further include a thread identifier.

6. The apparatus of claim 1, wherein the pipeline resource comprises a translation lookaside buffer.

7. The apparatus of claim 6, further comprising a filter coupled to the translation lookaside buffer to select at least one of the entries to be flushed.

8. A method comprising:

associating an address space identifier with a value;

hashing the address space identifier with a portion of the value; and

thereafter storing the value and the address space identifier in a pipeline resource.

9. The method of claim 8, further comprising storing the value and the address space identifier in an entry of the pipeline resource.

10. The method of claim 9, further comprising invalidating the entry if an update to the value occurs during a context.

11. The method of claim 10, further comprising selectively flushing the entry after invalidating the entry.

12. The method of claim 10, wherein invalidating the entry further comprises invalidating all non-global entries of the pipeline resource.

13. The method of claim 10, wherein invalidating the entry further comprises invalidating all entries of the pipeline resource associated with the address space identifier.

14. The method of claim 8, further comprising associating a second address space identifier with a second value; and

storing the second value and the second address space identifier in the pipeline resource.

16. A system comprising:

a processor including a pipeline resource including a plurality of entries each having one of a plurality of address spaces, each of the plurality of address spaces corresponding to one of a plurality of address space identifiers;

a hashing engine to hash one of the plurality of address space identifiers with a portion of a value to be stored in one of the entries; and

a dynamic random access memory coupled to the processor.

17. The system of claim 16, further comprising a control register coupled to the pipeline resource to provide the plurality of address space identifiers to the pipeline resource.

20. An article comprising a machine-readable storage medium containing instructions that if executed enable a system to:

associate an address space identifier with a value;

store the value and the address space identifier in an entry of a pipeline resource; and

flush a portion of the pipeline resource, the portion including the entry and other entries having the same address space identifier.

22. The article of claim 20, further comprising instructions that if executed enable the system to store a thread identifier in the entry.

23. The article of claim 20, further comprising instructions that if executed enable the system to associate a different address space identifier with a second value, the different address space identifier corresponding to a different active context than the address space identifier.

24. The article of claim 20, further comprising instructions that if executed enable the system to invalidate the entry if the value is updated during a context.

25. A method comprising:

providing a first address space identifier to a pipeline resource during a first context;

hashing the first address space identifier with at least a portion of a first data value; and

storing the first data value and the first address space identifier in a first entry of the pipeline resource.

27. The method of claim 25, further comprising invalidating the first entry if an update to the first data value occurs during the first context.

29. The method of claim 30, further comprising maintaining the first address space identifier in the first entry during the second context.

30. The method of claim 25, further comprising:

providing a second address space identifier to the pipeline resource during a second context; and

storing the second address space identifier in a second entry of the pipeline resource.

31. The system of claim 16, further comprising a filter coupled to the pipeline resource to flush the entries of one of the plurality of address spaces while the entries of the missing address spaces are maintained in the pipeline resource.

32. The article of claim 24, further comprising instructions that if executed enable the system to flush the portion of the pipeline resource on a next context switch after the invalidation.

33. The method of claim 27, further comprising flushing a portion of the pipeline resource associated with the first address space identifier after invalidating the first entry.

IX. EVIDENCE APPENDIX

There was no evidence submitted during prosecution.

X. RELATED PROCEEDINGS APPENDIX

There is no decision rendered by a court or the Board in the Appeal identified in Section II, above.